

Aztec C User Guide for the Amiga

**Version 5.0
October 1989**

**Copyright © 1988, 1989 by Manx Software Systems, Inc.
All Rights Reserved
Worldwide**

DISTRIBUTED BY:

**Manx Software Systems
P.O. Box 55
Shrewsbury, NJ 07702
(201) 542-2121**

USE RESTRICTIONS

You are permitted to install and use this product on a single computer. Multiple CPU systems require supplementary licenses.

Before using any Aztec C products, the License Registration included with this product must be signed and mailed to:

Manx Software Systems
P.O. Box 55
Shrewsbury, NJ 07702

COPYRIGHT

This software package and document are copyrighted ©1988, 1989 by Manx Software Systems. All rights reserved worldwide.

No part of this publication may be reproduced, transmitted, transcribed, stored in any retrieval system, or translated into any language without prior written permission of Manx Software Systems.

DISCLAIMER

Manx Software Systems makes no representations or warranties with respect to this product and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Manx Software Systems reserves the right to modify the programs and revise the contents of the manual without obligation to notify any person of such revision or changes.

TRADEMARKS

Aztec C, Manx AS, Manx LN, Z, and SDB are trademarks of Manx Software Systems. CP/M-86 and CP/M-80 are trademarks of Digital Research. MSDOS is a trademark of Microsoft. PCDOS is a trademark of IBM. UNIX is a trademark of AT&T Bell Laboratories. Macintosh and Apple II are trademarks of Apple Computer. Atari is a trademark of Atari Computers. Amiga is a trademark of Commodore-Amiga.

Manual Revision History

October 1989	Fifth Edition
December 1988	Fourth Edition
May 1988	Third Edition
April 1986	Second Edition
February 1986	First Edition

Table of Contents

Chapter 1 - Overview

Introduction	1 - 1
AZTEC C CONFIGURATIONS AND DOCUMENTATION	1 - 1
PORTABILITY, DEPENDABILITY, AND SPEED	1 - 3
How To Proceed	1 - 4
About This User Guide	1 - 5
Note to Previous Users	1 - 6
Aztec C Reference Manual	1 - 6
Aztec C Library Manual	1 - 6
Aztec C UniTools	1 - 7
Aztec C Source Level Debugger	1 - 7
Summary	1 - 7

Chapter 2 - Getting Started

System Requirements	2 - 1
Backup the Distribution Disks	2 - 2
Read the README File	2 - 2
Installation Procedure	2 - 3
System Overview	2 - 3
Basic elements of Aztec C	2 - 4
Types of Files	2 - 4
executable files	2 - 4
library files	2 - 4
header files include:	2 - 5
Library Files for Different Data Definitions	2 - 5

Environment Variables	2 - 7
Resident Libraries	2 - 7
Amiga Workbench	2 - 7
User Created Resident Libraries	2 - 8
#pragmas	2 - 8
Pre-compiled Headers	2 - 8
Debuggers - db sdb sdbf	2 - 9
Third Party Software	2 - 9
Freeware, Shareware, and Bulletin Boards	2 - 9
Multiple Segments and Transient Overlays	2 - 10
Mixed C and Assembly Code	2 - 10
QuikFix	2 - 10
QuikFix Compiler Options	2 - 11
QuikFix Z Editor Options and Commands	2 - 11
Batch Processing with QuikFix	2 - 12
Make and QuikFix	2 - 13
Further Information on QuikFix	2 - 13
SYSTEM PERFORMANCE AND COMPATIBILITY	2 - 13
MAKING A WORKING COPY OF AZTEC C	2 - 14
Libraries	2 - 17
Using Aztec C with a Hard Disk	2 - 19
If You Have A Problem	2 - 21

Chapter 3 - Tutorial

Development Environments	3 - 1
WORKING THROUGH THE TUTORIAL	3 - 1
Using the CLI	3 - 2
STARTING AZTEC C	3 - 2
A. Using floppy disks:	3 - 2
B. Using a hard disk:	3 - 2

COMPILING, ASSEMBLING, AND LINKING	3 - 6
ENVIRONMENT VARIABLES	3 - 7
THE RAM DISK	3 - 8
Summary	3 - 9

Chapter 4 - Technical Support

Have Everything With You	4 - 1
Know What Question You Wish To Ask	4 - 2
Isolate The Code That Caused The Problem	4 - 2
Use Your C Language Book And Technical Manuals First	4 - 2
When To Expect An Answer	4 - 3
Use Our Mail-in Service	4 - 3
Updates, Availability, Prices	4 - 3
Bulletin Board System	4 - 3
Phone Support	4 - 4
MANX PROBLEM REPORT	4 - 5
Description of problem	4 - 5

OVERVIEW

GETTING STARTED

AZTEC SHELL
TUTORIAL

TECHNICAL SUPPORT

OVERVIEW

1

Chapter 1 - Overview

Introduction

The Aztec C Software Development System is the ideal tool for both inexperienced and experienced C language programmers. Aztec C is powerful and flexible, offering the freedom to choose from a variety of programs and applications to fit your specific C programming needs.

Inexperienced users can employ the UNIX standard input/output routines to quickly create C programs that will run on the Amiga, without having to use the more complex Amiga Toolbox routines. *Experienced users* can employ extended interfaces with full access to all of the Amiga Workbench and other ROM routines.

Aztec C is fully compatible with the draft proposed ANSI standard and provides extensive support for 68020, 68881, device drivers, Intuition, and other specialized Amiga software and hardware. Numerous examples are provided with your Aztec C package.

AZTEC C CONFIGURATIONS AND DOCUMENTATION

The Aztec C product line for the Amiga offers two core systems and several add-on products, as follows:

- **Aztec C - Professional System**

The Professional level of Aztec C for the Amiga includes the Aztec Shell, Compiler, 68000 Macro Assembler, Overlay Linker, Librarian, Run Time Libraries, Profiler, Full Amiga Workbench Interface, Z editor, and a Portable C Library Interface.

Aztec C offers four different IEEE floating point formats: Manx IEEE double precision emulation, Motorola 68881 coprocessor, Motorola IEEE Floating Point, and Amiga Fast Floating Point. These formats are especially helpful with math programming.

- **Aztec C - Developer System**

The Developer level of Aztec C for the Amiga includes the Aztec Shell, Compiler, 68000 Macro Assembler, Overlay Linker, Source Level Debugger, Librarian, Run Time Libraries, Profiler, Full Amiga Workbench Interface, a Portable C Library Interface, and the complete set of Manx "Unitools."

UniTools is a productivity enhancement package which includes the utilities **make**, **diff**, **grep**, and **z** editor.

make is a program maintenance utility

diff is a source file comparison utility

grep is a pattern-matching program

z is a powerful text editor, similar to the
UNIX vi editor.

Documentation for both the Professional System and the Developer System includes: the *Aztec C User Guide*, the *Aztec C Reference Manual*, the *Aztec C Library Manual*, and *Aztec C UniTools* for the Amiga. Note that if you have purchased the Professional System, you will receive the *documentation* for UniTools, but only receive the Z editor portion of the software. **make**, **grep**, and **diff** are not included in the Professional System.

The special Aztec C add-on products include:

- **Source Level Debugger**

SDB is Manx's critically acclaimed Source Level Debugger. It is an interactive debugger, designed for fast response and ease in debugging. The windows in SDB display C source and command output separately, with a third window for entering commands. With SDB, you can...

- ...display all active function names*
- ...display values of passed parameters*
- ...examine variables from any active function*
- ...use function or line-by-line tracing*
- ...set breakpoints by lines, functions, or variables*
- ...see actual C source as it executes*
- ...customize the debugging environment with reusable command macros and procedures*
- ...and more!*

Documentation for SDB is the *Aztec SDB Guide*.

- **Aztec Library Source**

Aztec Library Source includes the original assembly and C source to all routines found in the Aztec Run Time Libraries. Aztec C Library Source does not come with a separate manual; all of the information needed to use Library Source is found in the *Aztec C Library Manual*, which is included with the basic Aztec C package.

PORTABILITY, DEPENDABILITY, AND SPEED

Manx makes several types of Aztec C Software Development Systems. Manx's *native development systems* allow development to be done on the Amiga; Manx's *cross-development systems* allow development to be done on other machines, with the resulting programs downloaded to the Amiga.

Manx also has compilers for developing C programs which run on systems other than the Amiga. Native development or cross development systems

are available for the Macintosh, Atari, Apple II, MS-DOS, and other popular systems.

When you chose Aztec C for the Amiga, you chose **dependability**. Since 1981, Manx has been acquiring and developing an extensive and comprehensive set of test suites that thoroughly exercise our products to provide the high degree of reliability and dependability that our users expect. Thousands of users have worked with Aztec C for the Amiga since 1986 and have given us their feedback—good and bad. Manx uses these suggestions to continually improve and enhance Aztec C.

Aztec C is also known for its **speed**. When you use the Aztec compiler, you can specify precompiled header files, thereby speeding up compilation time. If you already have a header file of your own, our compiler allows you to build precompiled header files and use them with our package.

The optimizing assembler is another important factor when you are concerned about the size and speed of your program. It performs branch shortening, as well as other code improvements.

How To Proceed

Now that you have Aztec C, the first thing you are going to do is...

...read the manual?

If you are like most users, probably not. The Manx manuals are very thorough—and a very important part of your Aztec C package—but we recognize that you may be anxious to put Aztec C to work as soon as you can. To some users, that means “skimming” the important parts and getting to the rest later.

In light of this, we would like to offer a bit of advice. The list below, for both new and experienced users, shows how we suggest you proceed. This may help you get off to a quicker start.

- ❑ Step 1: Read the **readme** file (on disk).
- ❑ Step 2: Install your disks, using the installation instructions in Chapter 2, **Getting Started**.

- ❑ Step 3: Familiarize yourself with Aztec C by completing this chapter. Be sure to work through the "hello world" program which Aztec provides step by step in Chapter 3, **Tutorial**.
- ❑ Step 4: Look through both your *Aztec C Reference Manual* and your *Aztec C Library Manual*. Pay particular attention to the **Language Specifications**, **Compiler**, **Assembler**, and **Linker** chapters of the *Reference Manual*. Even if you are an experienced Aztec C user, you should skim through these chapters for the most up-to-date information on Aztec C.
- ❑ Step 5: If you have the Developer System, refer to your *Aztec C UniTools manual* for detailed information on **diff**, **grep**, **make**, and the **z** editor, and your *Aztec C Source Level Debugger Manual* for detailed information on **sdb**.

About This User Guide

This guide gives you the basic information that you need to start working with Aztec C. It consists of this introduction, a guide for getting started, a tutorial, and information on our technical support.

Of course, this manual also includes a detailed **Table of Contents** and a thorough **Index**.

Throughout this manual, we use the following conventions:

- | | |
|---------------------|---|
| input | to indicate data entered by the user (e.g., commands, options, and functions) |
| output | to show text that is generated by the computer. |
| DEFINITION | small uppercase bold is used on terms that may be new to the user; most likely will include explanation or definition of term |
| {choice1 choice2} | braces and a vertical bar mean that you have a choice between two or more items |

placeholders

information that must be supplied by the user, for example, *filename*, *range*, *identifier*, etc.

[*optional*]

brackets to show optional information.

Note to Previous Users

If you have used Aztec C for the Amiga prior to the release of Version 5.0, then you should be aware of the many changes and additions that have been implemented. The following paragraphs discuss these changes and direct you to the appropriate chapters for further information.

Aztec C Reference Manual

The most important change to Aztec C for the Amiga is that it is now *fully draft-proposed ANSI standard*. Aztec C's implementation of ANSI C is discussed fully in Chapter 2 of your *Aztec C Reference Manual, Language Specifications*.

It is very important that you read Chapter 3, **Compiler**, particularly with regards to all of the compiler options. This chapter discusses many substantial changes to the compiler.

The following chapters also include changes from previous releases and should be reviewed:

- **Assembler**
- **Utilities**
- **Debugger**
- **Technical Information**
- **Error Messages**

Aztec C Library Manual

The library functions included with your Aztec C system have changed substantially with the draft-proposed ANSI standard. Both the **Library**

Overview chapter and the **Library Functions** chapter reflect these changes and should be reviewed.

You will notice, also, that the **Library Functions** chapter has been improved and expanded, offering a complete synopsis and description of each function. Most functions are listed separately, with frequent references to other related functions.

Aztec C UniTools

With the **make** utility, you will find one minor change: Default rules, when applied to files in other directories, will place the results in those other directories. Previously, the results were placed in the current directory.

Aztec C Source Level Debugger

Refer to Chapter 2, **Getting Started**, and Chapter 4, **Commands**, for changes and new commands.

Summary

You should find that the extended type-checking capabilities of ANSI C make programming easier. And, since ANSI is the standard for the C language, your Aztec C programs will now be more portable than ever before.

Chapter 2 - Getting Started

This chapter provides the information you need to begin using Aztec C. The following are the major topics:

- System requirements
- Backing up the distribution disks
- The **readme** file
- Installation Procedure
- System overview
- QuikFix Feature
- System Performance & Compatibility

System Requirements

To efficiently use Aztec C, make sure your hardware and software meet the following specifications:

- Aztec C is compatible with the Amiga 500, Amiga 1000, Amiga 2000, and Amiga 2500. Aztec C is also compatible with all known accelerator boards. Manx recommends at least one floppy disk drive, preferably two. Aztec C is easily installed on a hard disk. Aztec C will run comfortably in 500k of memory.
- SDB, the Source Level Debugger, only uses 90k, but requires a minimum of 256K of available RAM memory. (sdb is included as part of the Aztec C68k Amiga Developer System)

Backup the Distribution Disks

The `diskcopy` program provided with your Amiga will copy all of the distribution disks, or you can use any other utility that copies disks. The disks are not copy protected. It is an excellent idea to leave the original distribution disks write-protected and to make at least one backup copy. If you have a problem with any of the distribution disks, call or FAX Manx Customer Support. The Fax number is 201-542-8386 and the Customer Support number is 201-389-0290.

Walt Disney once said, "We don't make movies to make more money. We make money to make more movies." There are estimates that for every legal copy of commercial software there are three illegal copies. Bootleg software, in general, diminishes the quantity and quality of software available for any machine. Giving away copies is as wrong as selling copies. Support better software by buying it and encouraging others to do so.

Read the README File

You can access the `read.me` file from the CLI environment by inserting distribution disk #1 into drive `df0:` and entering,

```
execute df0:readme
```

The `readme` execute file is one word and there is no ".". The actual data file has a "." between "read" and "me".

The `read.me` document provides corrections and additions to information in the manual. Some information will only be found in the `read.me` file. It is important to read this file.

If you want to send the **read.me** document to the printer, use the command:

```
type > PRT: read.me
```

Installation Procedure

The Aztec C system must be installed to a floppy disk or hard disk. To install Aztec C from the CLI environment insert your distribution disk labeled #1 into drive **df0:** and enter

```
execute df0:install
```

The installation procedure will walk you through the steps to create a floppy or hard disk working version of Aztec C.

It may be useful to read the **System Overview** Section below, and the rest of this chapter before performing the installation. A knowledge of the basic system structure and interactions can be useful in executing the install procedure.

The installation procedure will create a default system or a customized higher performance system. If you are new to Aztec C, it is advisable to use the default system. The default system sacrifices some code and execution efficiency, but it chooses options that require the least amount of effort or knowledge. If you require faster, smaller code and faster compile time, then choose the customized version. The system configuration can be changed later by re-installation. The system configuration can also be changed by changing options to the Aztec programs, by changing libraries, or by setting different values for environment variables.

A description of all the files on the distribution disks will be found in the **read.me** file on distribution disk #1.

System Overview

The Aztec C System is highly configurable. Features and characteristics can be enabled, disabled, or changed by user definable options. Some features may also require changing libraries or header files.

These options give you enormous flexibility. They also require careful definition. It is important to understand all of what is required for a

particular feature. Most of the information needed is provided in this manual. Some of the information you need may be in the Amiga system manuals.

The following overview describes features and structures that are useful to know to install, modify, and use the Aztec C System.

Basic elements of Aztec C

Types of Files

executable files	no filename extension
library files	.lib filename extension
header files	.h filename extension

executable files

programs

- cc** - compiler
- as** - assembler
- ln** - linker
- sdb** - source level debugger
- z** - source editor
- make** - project administrator

commands

- set** - set environment variables

utilities

- lb** - object librarian
- makefd** - resident library utility
- cnm** - object utility
- ord** - object utility
- diff** - comparison utility
- grep** - pattern search utility

library files

- math functions - **m.lib**
- UNIX, ANSI, and general functions - **c.lib**
- Screen Functions - **s.lib**
- Amiga ROM kernal functions - see **read.me**

header files include:

- the complete standard ANSI headers
- standard Aztec headers
 - unbuffered (UNIX) I/O - **fcntl.h**
 - ROM kernel function prototypes - **functions.h**
 - Inline ROM kernel function calls - **functions.h**
- the complete standard Amiga headers

Library Files for Different Data Definitions

Aztec C supports a number of different data definitions. It would be convenient to have just one set of definitions, but different performance and compatibility requirements make it necessary to provide alternatives. Most non-commercial users and many commercial users will find the default definitions are more than adequate. The data definition in effect is determined by compiler options and by the libraries used to link a program.

The choices marked below with (!) usually produce the smallest and/or fastest code. The items marked with (d) are the system defaults.

The guidelines for choosing system defaults are to provide the configuration that requires the least amount of work and knowledge.

The basic variations for data definitions are:**int definition**

- 16 bit (!)
- 32 bit (d)

data and code reference definition

- small code and data (!d)
- large code and data

floating point definition

- Manx IEEE Floating Point (d)
- Amiga IEEE Floating Point
- Motorola Fast Floating Point (!)
- 68881 Floating Point (!)

Notice that 16 bit ints produce smaller faster code, but 32 bit ints are the system default. Why not just default to 16 bit ints? The reason is that Amiga supplied routines are based on 32 bit ints. The use of 16 bit ints requires special attention when calling Amiga routines. Either the Amiga

parameters that require 32 bit data items must be declared as longs or the **functions.h** header file must be included. Likewise, header files are required for calling some Manx Aztec routines if 16 bit ints are used. To make Aztec C easier to use, the least complicated configuration is offered as the default. Experienced users who want the increased speed of 16 bit ints or some other system enhancements can provide the necessary option to the compiler, set up different libraries for linking, and include the necessary header (.h) files in their programs.

The three basic Aztec libraries are **c.lib**, **m.lib**, and **s.lib**. These libraries assume 32 bit ints, small code and data, and Manx IEEE floating point.

The complete set of **c.libs** is

c.lib	32 bit ints	small code and data
c16.lib	16 bit ints	small code and data
cl.lib	32 bit ints	large code and data
cl16.lib	16 bit ints	large code and data

The complete set of **s.libs** is: **s.lib**, **s16.lib**, **sl.lib**, and **sl16.lib**.

The four basic floating point libraries are:

m.lib	Manx IEEE Float
ma.lib	Amiga IEEE Float
mf.lib	Motorola Fast Floating Point (FFP)
m8.lib	68881 Float

Each of these libraries have four variations:

m.lib , ma.lib , mf.lib , m8.lib	32 bit ints and small code/data
ml.lib , mal.lib , mfl.lib , m8l.lib	32 bit ints and large code/data
m16.lib , ma16.lib , mf16.lib , m816.lib	16 bit ints and smallcode/ data
ml16.lib , mal16.lib , mfl16.lib , and m8l16.lib	16 bit ints and large code/ data

So, as an example, if 68881 floats were used with 16 bit ints in large code and data mode, the linker libraries would be **cl16.lib** and **m8l16.lib**. The compiler options would be: **-ps**, **-mcd**, and **-f8**. If screen functions were used then the **sl16.lib** library would also be needed.

Environment Variables

Manx Aztec C supports four environment variables. These variables are set using the Manx **set** command. The Amiga CLI environment also has a **set** command. The two commands at the time of this writing are not compatible. One way of avoiding confusion would be to rename the Aztec **set** command to **azset** or some other unique name. The **read.me** file may contain more recent information on this issue.

More detailed information on environment variables will be found in the *Aztec C Reference Manual* and in the *Aztec C Unitools Manual*.

The four environment variables supported by Aztec C and their associated commands are as follows:

- CCOPTS - default compiler options (compiler - **cc**)
- CCEDIT - quick fix command line (compiler - **cc** editor - **z**)
- INCLUDE - search path(s) for header files (compiler - **cc**)
- CLIB - search path(s) for libraries (linker - **ln**)

Resident Libraries

Amiga Workbench

The Amiga Workbench routines are all accessed through the resident library mechanism. Unlike other library routines that are linked in with a program, resident library routines are accessed through a dynamic mechanism at run time. There are two ways that a program written in Manx Aztec C can reference the Workbench resident library routines, by the use of glue routines or through inline function calls.

A complete set of glue routines for the Amiga Workbench is included in each of the **c.lib** libraries. So, when an Amiga Workbench routine is called the appropriate glue routine is executed to set up the correct environment for the call to the resident routine.

A faster way to access Amiga Workbench routines is by using inline calls. If the header file **functions.h** is included in a program by coding,

```
#include <functions.h>
```

then, instead of generating code to call a glue routine, the compiler will generate an inline direct call to the Amiga Workbench routine.

For more information on resident library support see the *Aztec C Reference Manual* **Compiler** Chapter. For information on Workbench routines see the *Aztec C Library Manual*.

User Created Resident Libraries

You can create your own resident library routines. Instructions for creating them can be found in the **Compiler** Chapter of the *Aztec C Reference Manual*.

#pragmas

The creation of inline calls to resident library routines is implemented by use of the ANSI feature, **#pragma**. The **Compiler** Chapter of the *Aztec C Reference Manual* discusses the pragmas provided as part of Aztec C. It includes pragmas for resident library calls, registerized function calls, and interrupt handler definition.

Pre-compiled Headers

Header files contain definitions for structures, prototypes, and pragmas. There are also **#define**, **typedef**, and **enum** statements. Aztec C headers contain every kind of C statement except those that produce code.

It is typical that a very small percentage of a header file is used by any one program. The header **functions.h**, for example, contains information for over 500 Workbench calls, but most programs use only a small fraction of them. It is highly convenient, however, to put all of the calls into one header file. The penalty is that every time **functions.h** is included the entire contents is processed by the compiler. To maintain the convenience of putting everything into one file but to reduce the overhead of compilation, Manx Aztec C provides for pre-compiled header files.

To make a pre-compiled header file, create a source file with a **#include** statement for each header to be pre-compiled. The compiler is then invoked with the **-hi** option. It is important that any options that effect data definition also be specified. The data definitions of a pre-compiled header must match the data definition for any compilation that uses it. The compiler processes the headers and writes the compiled table information out to a file. Later this file is read in by the compiler by specification of the **-hi** option. Since all of the information is in compiled form, the overhead is simply that of the I/O to read the data. Pre-compiled headers can save a significant amount of time.

A full description of pre-compiled headers can be found in the **Compiler** Chapter of the *Aztec C Reference Manual*.

Debuggers - db sdb sdbf

Aztec C for the Amiga has three debuggers, **db**, **sdb**, and **sdbf**. **db** is an assembly language debugger and is described in the *Aztec C Reference Manual*. **sdb** and **sdbf** are C source level debuggers. **sdb** is described in the *Aztec C Source Debugger Manual*. **sdbf** is identical to **sdb** except that it supports Motorola Fast Floating Point.

Third Party Software

There are a number of third party software packages, such as Power Windows and Amiga Lint, that extend the capabilities of Aztec C. Call the Manx sales department for a complete list and current prices of third party software.

Freeware, Shareware, and Bulletin Boards

There is a large body of public domain software for the Amiga. Some freeware/shareware is included on the Manx Aztec Distribution disks. Some public domain software can be found on the Manx bulletin board.

Another good place to look is BIX. Information on joining BIX can be found in *BYTE* magazine.

The Compuserve Amiga conference is accessed by entering,

```
go amigatech
```

At the time of this writing, *Amazing Computing* was publishing a comprehensive list of the AMICUS and Fred Fish disks in each issue. Copies of the disks are provided for a nominal fee. Their number is 1-800-345-3360.

Another source of public domain software would be your local Amiga User Group.

Multiple Segments and Transient Overlays

Manx Aztec C supports the creation of multiple segments. Multiple segments can be scatter loaded. Scatter loading gets around memory fragmentation problems. Multiple segments can also be dynamically loaded as transient overlays. Transient overlays allow very large programs to run in a small memory space. More detailed information on this feature can be found in the **Technical Information** and **Linker** Chapters in the *Aztec C Reference Manual*.

Mixed C and Assembly Code

Aztec C offers three levels of assembly support: inline, intermediate, and separate.

Inline assembly support allows 680x0 assembly statements to be mixed directly with C code. The general form for inline assembly is:

```
c code  
  
#asm  
    assembly code  
#endasm  
  
c code
```

The output of the Aztec C compiler is assembly code. Usually the assembler is invoked quietly by the compiler, and when the assembler finishes the intermediate assembly code is deleted. You can, however, save the intermediate assembly, modify the code, and invoke the assembler directly.

Entire assembly level source files can be created, separately assembled, and then combined with compiled C modules.

More detailed information on this topic can be found in the *Aztec C Reference Manual* in the **Assembler**, **Compiler**, **Linker**, and **Technical Information** Chapters.

QuikFix

QuikFix is a new facility with version 5.0 of Aztec C. **QuikFix** is an interactive interface between the Aztec C compiler and the Z editor, or any

editor that supports AREXX (see the `read.me` file). The compiler is set in QuikFix mode by specifying the `-qf` option. The command that is activated if errors are encountered is defined by the CCEDIT environment variable.

QuikFix Compiler Options

The `-qf` option tells the compiler to activate the QuikFix environment. For example:

```
cc -qf hello
```

will compile `hello.c` and, if there are any errors, execute the CCEDIT command string.

When QuikFix is activated, the compiler writes specially formatted error messages to a file called:

AztecC.Err

If errors are encountered during compilation, the compiler will execute the contents of the CCEDIT environment variable as an AmigaDOS command. When the `z` command finishes with a zero value, the compiler recompiles the program using the original options. If a non-zero code is returned to the compiler the compiler terminates with an error.

A convenient way to use QuikFix is to specify the `-qf` option in the CCOPTS environment variable. For instance, if you enter:

```
set CCOPTS=-qf
```

in the CLI environment, everytime the compiler is run it will run with the `-qf` option appended to the beginning of the option list.

QuikFix Z Editor Options and Commands

The basic CCEDIT specification to invoke `z` as part of the QuikFix facility is,

```
set "CCEDIT=z -e"
```

The quotes are mandatory. This string tells the compiler to invoke `z` if the `-qf` option was specified. The `-e` option tells `z` that it was invoked as part of the QuikFix facility and, more specifically, to issue the `:cf` command. The `:cf` command causes `z` to read the `AztecC.Err` file. The `AztecC.Err` file has information that indicates the appropriate source file. After reading this

source file, **z** positions the cursor to the line of the first error. At this point, **z** will also display the compiler error message associated with this line.

Three **z** commands are provided to move from error line to error line. They are,

:cp	go to previous error
:cc	redisplay current error
:cn	got to next error

The **Z Editor** Chapter in the *Aztec C UniTools Manual* describes how to assign function keys for **z** via the **z.opt** file. Assigning '**:cp^M**', '**:cc^M**', '**:cn^M**' to three different function keys can save some time in going through files with a large number of errors.

If lines are added to or deleted from the source file, it will effect the error positioning commands. It is a good idea to review the lines in error before making changes and, if it is practical, to make corrections working backwards so that lines added and deleted do not affect the error line positioning commands.

The editor will only process 25 errors at a time. the **:cf** command will read in the next 25 errors.

If **z** is exited via **:q** or **:wq** the compiler will recompile the modified code using the original options. If **z** is exited via **:cq** then the compiler will terminate with an error status.

Batch Processing with QuikFix

If the compiler is invoked with the **-wq** option, the **AztecC.Err** file will be appended instead of rewritten. This allows multiple compilations to be run with the errors from each written into one **AztecC.Err** file. With the **-wq** option, a command specified through **CCEDIT** is *not* executed.

By invoking **z** and then issuing the **:cf** command repetitively you can step from file to file correcting errors. You do not need to know the names of the source files.

Invoking **z** with the **-e** option is equivalent to invoking **z** and immediately issuing the **:cf** command.

Make and QuikFix

One of the great features of **QuikFix** is that it can be used in combination with the **make** utility.

To use **QuikFix** interactively with **make** set up **CCOPTS** to include the **-qf** option or specify **-qf** with the **make CFLAGS** macro. Then set the **CCEDIT** environment variable to "**CCEDIT=z -e**". Then when you use **make**, **z** will be invoked if there are any source errors during compilation. If the error is corrected, the compiler will recompile the source and **make** will then proceed to any additional files that need to be compiled.

Normally **make** terminates if there is an error in compilation. With **QuikFix** **make** will only terminate if the **z** editor is exited with the **:cq** command.

Further Information on QuikFix

The **QuikFix** options are documented in the **Z Editor** Chapter of the *Aztec C Unitools Manual* and in the **Compiler** Chapter of the *Aztec C Reference Manual*.

System Performance and Compatibility

The following is a brief discussion of the most significant Aztec C options that effect system performanc.

Running the compiler in small code and small data mode in combination with 16 bit **ints**, produces smaller faster code. The compiler options to produce this result are,

-mcd -ps

The libraries to specify at link time are **c16.lib**, **m16.lib**, and **s16lib**.

When 16 bit **ints** are used, the **functions.h** header file should be included if Amiga Workbench routines are called. For other library calls, the *Aztec C Library Manual* should be consulted. This manual specifies the header file (**.h**) that should be included for each library routine. Each header covers multiple routines. It is suggested that you use pre-compiled headers. Pre-compiled headers are discussed earlier in this chapter and in the **Compiler** Chapter in the *Aztec C Reference Manual*.

To produce even smaller faster code add the **-so** option,

-mcd -ps -so

The compiler option, **-so**, causes the compiler to perform maximum code optimization. It is sometimes difficult to debug with this option on since code is rearranged. Some users will run with this option off until the last stage of development. Others run with it on as a matter of course.

To produce faster floating point code, Motorola Fast Floating Point (FFP) or 68881 floating point options can be used. With Motorola FFP format, floats and doubles are 32 bits as opposed to the 64 bits for IEEE format. Thus, the precision is less, but for many applications it is sufficient. The compiler option,

-ff

must be added to the compiler option list for FFP and the **mf.lib** libraries are used for linking. Make sure that you use the right **mf** library. If 16 bit ints are used, for instance, it is **mf16.lib**. To use the 68881 floating point routines, the target machine must have a 68881 co-processor, the compiler option is **-f8**, and the library is one in the **m8** family. The different floating point libraries are discussed earlier in this chapter.

If you are including a large header file in your program, or a large number of smaller header files, compilation time can be speeded up by precompiling some header files. Pre-compiled headers are discussed in the **Overview** Section of this chapter.

The fastest executing software floating point in this package is the Motorola Fast Floating point. If you have a 68881 chip then the fastest floating point is the 68881 floating point.

Making a Working Copy of Aztec C

The #1 distribution disk has an installation procedure that is executed by placing the disk in drive **df0:** and entering,

```
execute df0:install
```

This is the preferred installation method.

The following is provided for those who may want to customize the installation beyond what is possible with the install procedure.

1. Make sure that the write protect tabs on the master disks are in the no-write position.

Turn on your machine. If you own an Amiga 1000, insert the KickStart disk and wait until it displays the Insert WorkBench screen, and then insert your standard Workbench boot disk. If you do not own an Amiga 1000, the Kickstart disk is not required, and you can boot with just your Workbench disk.

2. From the Workbench, copy your **sys1:** disk to a blank formatted work disk, then store your original **sys1:** disk in a safe place.

3. You should now organize your disks in a manner which will allow you to comfortably develop applications. How you organize your disks will depend primarily on whether you have one or two disk drives, and also on the amount of memory that is available.

A. Two drives and 1 MB or more of memory available.

The #1 distribution disk has an installation procedure that is executed by placing the disk in drive **df0:** and entering,

```
execute df0:install
```

This is the preferred installation method. The following is provided for those who may want to customize the installation beyond what is possible with the install procedure.

Your best approach to setting up this type of system would be to have two working disks which you regularly boot from for C development, and then a third disk which is used to hold your actual C programs.

- Your first disk, which would reside in **df0:** at all times, would be a simple copy of the **sys1:** disk.
- The second disk would contain all utilities, fonts, libraries, etc., that you want to load into the RAM disk.

This may require copying certain directories from your **sys1:** disk to your second working disk. In addition, you should copy certain CLI commands that are not provided on the Aztec disks, such as Format and Diskcopy, from your own Workbench disk to either the first or second boot disk.

For example, if you wanted to have your Amiga libraries, Aztec libraries, and CLI commands loaded into RAM, the portion of your Startup-Sequence that would accomplish this would be:

```
copy df1:libs ram: NUL
assign LIBS: ram:
copy df1:c ram: NUL
path ram:
copy df1:lib ram:
set CLIB=ram:
```

All utilities, include files, etc., that you do not want to load into RAM (or that will not fit in RAM) should reside on the disk which will be in **df0:**. After booting, you can replace the **df1:** disk with your Aztec work disk, which contains all of your C and object files. The basic idea is to have as many commands in RAM as possible, so that both **df0:** and **df1:** can be used for disk copying, C development, etc., instead of for holding system fonts and commands.

B. Two drives and less than 1MB of memory available.

The #1 distribution disk has an installation procedure that is executed by placing the disk in drive **df0:** and entering,

```
execute df0:install
```

This is the preferred installation method. The following is provided for those who may want to customize the installation beyond what is possible with the install procedure.

If you have two floppy drives available, you should set up your first disk to be the Aztec C development disk, containing all necessary files including the compiler, assembler, etc. The second disk should hold all of your object modules, C source code, etc.

You should first decide which files you will need on disk 1. The directories you will need to modify include the C directory, **bin** directory, **devs** directory, **lib** directory, and **libs** directory.

- In the C directory, you should delete any CLI commands which you do not plan to use, such as Search and Edit. **Be sure not to delete the Run command, as it is required by the CLI.** Also, you should copy the Format and Diskcopy commands from your Workbench disk if you need them, since the Aztec distribution diskettes do not include these commands.
- In the devs directory, you should delete any devices which you do not plan to use, such as **narrator.device**, (used for speech synthesis) and **printer.device**.
- In the libs directory, you can delete the **translator.library**, which is used by the speech synthesizer. If you are not using transcendental functions, the **mathtrans.library** can be deleted; if you are not using floating point math in your programs, the **mathieedoubbas.library** can be deleted. You may also delete **info.library** if you do not plan to use the Workbench in your development work.

Libraries

The C libraries you plan to use will depend upon several factors; only the libraries you intend to use should be included on disk 1. First, if you do not plan to use floating point math, delete the **m.lib** library from disk 1, as it will not be needed. Next, make sure *only* the versions of **c.lib** and **m.lib** that you require (if you need them at all) are on the disk.

The libraries that use large code/large data have the letter "l" in their names, and libraries that use 16 bit integers have a "16" in their names. These two types can also be combined. 32 bit ints or small code/small data libraries have no additional letters. As an example, the large code/large data, 16 bit integer c library is **cl16.lib**. The small code/small data, 32 bit c library is **c.lib**.

There are four types of math libraries, distinguished by the characters included in their names (shown in parenthesis):

- Amiga IEEE floating point emulation (64-bit) libraries (**ma**)
- Motorola Fast Floating Point math (32-bit) libraries (**mf**)
- Manx IEEE floating point functions (64-bit) (**m**)
- The 68881 numeric coprocessor libraries (64-bit) (**m8**)

As an example, the large code/ large data 32 bit int Amiga IEEE emulation library is called **mal.lib**. Note that most of the libraries are contained on the **sys2:** disk in the **lib** directory and should be copied to **sys1:** if necessary.

C. One floppy drive.

The #1 distribution disk has an installation procedure that is executed by placing the disk in drive **df0:** and entering,

```
execute df0:install
```

This is the preferred installation method. The following is provided for those who may want to customize the installation beyond what is possible with the install procedure.

If you have a one-floppy system, your system will be based on a trimmed-down version of the **sys1:** disk.

- From the **devs** directory, delete **narrator.device**, and possibly also **printer.device** if you do not need to use a printer.
- From the **libs** directory delete the **translator.library** file. If you are not using transcendental functions in your programs, you can delete **mathtrans.library**; if you are not using floating point, you can delete the **mathieeedoubbas.library**.
- In the **lib** directory, you should keep only the libraries which you need. See the section **Libraries**, above, for details on the libraries that are available.
- From the **C** directory, many commands can be deleted. Notably, the **Edit** editor and the **search** command should be deleted. The only command you absolutely cannot delete from the **C** direc-

tory is the **Run** command, which is needed by the CLI. You should also copy the **format** and **diskcopy** commands from your existing Workbench disk, as these are not present on the Aztec distribution diskettes.

D. Two disk system quick installation.

The #1 distribution disk has an installation procedure that is executed by placing the disk in drive **df0:** and entering,

```
execute df0:install
```

This is the preferred installation method. The following is provided for those who may want to customize the installation beyond what is possible with the install procedure.

If you have two floppy drives and wish to get up and running right away, you can accomplish this by making duplicates of the first two Aztec disks (**sys1:** and **sys2:**). You then must delete all unnecessary libraries from the **lib** directory of **sys2:**, as well as the entire **lint** directory (assuming you have not bought a commercial lint program). See the section **Libraries**, above, for details on the libraries that are available.

Then boot with the copy of **sys1:** in **df0:** and with the copy of **sys2:** in **df1:**. If you plan to follow the tutorial, you should also create a directory called **Aztec** on the **sys2:** disk using the CLI command **makedir**:

```
cd df1:
```

```
makedir Aztec:
```

This will provide a simple working environment for you to start with. If you find this setup too slow and wish to use the RAM drive, or simply need instructions on how to optimize the basic setup, see the instructions in sections **A** or **B**, above.

Using Aztec C with a Hard Disk

The #1 distribution disk has an installation procedure that is executed by placing the disk in drive **df0:** and entering,

```
execute df0:install
```

This is the preferred installation method. The following is provided for those who may want to customize the installation beyond what is possible with the install procedure.

If you have a hard disk, you should install Aztec C onto the hard disk rather than work from floppies. Hard disk installation is fairly straightforward, and consists of the following steps:

A. Enter the system directory on your hard disk and click on the CLI. If it is not there, you may have to start the CLI from your boot floppy. If the CLI is also not on the boot floppy, you will have to run the **Preferences** program to turn on the CLI. See your *Amiga User's Manual* for details on the **Preferences** program.

B. Ensure that you are at the root directory of your hard disk, create a directory named **Aztec** at the root level, and enter the **Aztec** directory :

```
cd dh0:
mkdir Aztec
cd Aztec
```

C. Insert your **sys1:** disk into the **df0 :** drive and type:

```
mkdir bin
copy df0:bin dh0:Aztec/bin
```

to copy the **bin** directory from **sys1:** to your hard disk. Next, copy the **include** directory:

```
mkdir include
copy df0:include dh0:aztec/include ALL
```

The **ALL** option for **copy** specifies that all subdirectories under the **include** directory should be recreated on the destination. You should now copy a program called **set** from the **C** directory on **sys1:** to the logical **c :** device:

```
copy df0:c/set c:
```

The **set** command will be used later in conjunction with the **Startup-Sequence**.

Now copy the `lib` directory:

```
makedir lib
copy df0:lib dh0:aztec/lib
```

D. The `sys1:` disk has now been successfully copied. Remove `sys1:` and put `sys2:` into `df1`. Copy the `bin` and `lib` directories from the `sys2:` disk in the same manner that you did for `sys1:`, but do not create any new directories.

E. Remove the `sys2:` disk and insert the `sys3:` disk. Copy its `bin` and `lib` folders as you did for the `sys2:` disk.

F. You now have to modify your Startup-Sequence so that it properly sets up your system for use with the Aztec C system. If you boot your hard disk system from a floppy, you should insert that floppy now and type:

```
cd df0:s
```

If you have an autoboot drive, simply type:

```
cd s:
```

G. You should now enter your editor of choice and load in the file Startup-Sequence. This is a file which is executed when you boot your machine; it performs general setup work. At the end of the file you should add the following:

```
path add dh0:Aztec/bin
set CLIB=dh0:Aztec/lib
set INCLUDE=dh0:Aztec/include
```

If You Have A Problem

If your Aztec C system is not functioning, repeat the installation procedures for your particular hardware/memory setup, paying close attention to each step.

If your Aztec C system is still not functioning, contact Technical Support. See Chapter 4, **Technical Support**, for more information.

Chapter 3 - Tutorial

This chapter presents a basic tutorial of Aztec C for the Amiga. For this tutorial, it is assumed that you have already followed all of the procedures outlined in Chapter 2, **Getting Started**, and that you are now ready to begin using Aztec C.

This tutorial is meant only to present you with the basics of using the Aztec system; you will be referred to other chapters that will give you more detailed information about the topics discussed here.

Development Environments

With Aztec C, the default, or boot, environment is the **CLI**, however there are several public domain shells available which you may want to use in place of CLI, which will work fine with Aztec C.

WORKING THROUGH THE TUTORIAL

This tutorial introduces the CLI and briefly describes some of the more important commands you need to know, including how to organize and move around among your files; how to enter a program; how to compile, assemble, and link your programs; how to set environment variables; and how to take advantage of the Amiga RAM disk.

Using the CLI

STARTING AZTEC C

A. Using floppy disks:

If your machine is not on, turn it on. If you have an Amiga 1000, insert the KickStart disk and wait until it displays the Insert WorkBench screen, but do not insert the WorkBench disk.

If your machine is already on, you must perform a warm boot by holding down the control key and pressing the two Amiga keys simultaneously. After the boot, you should see the Insert WorkBench screen, but do not insert the WorkBench disk.

Instead of your WorkBench disk, insert the Aztec C disk labeled number 1. (Use your working copy--not the original disk!) This disk boots directly into the CLI and will display the Aztec startup message.

B. Using a hard disk:

Boot your hard disk in the usual manner.

- If your normal boot places you into CLI, type

```
cd dh0:aztec
```

to invoke Aztec C.

- If your normal boot does *not* place you into CLI, then you must go into the system directory and look for the CLI icon.

If there is a CLI icon in your system directory, then double click on it to enter CLI.

If there is *not* a CLI icon in your system directory, exit the system directory, which will place you back at the root level. Double click on the Preferences icon, turn on CLI, save, then exit Preferences. Go into the system directory; you should now see the CLI icon. Double click on this icon to enter CLI.

Once you are in CLI, type

```
cd dh0:aztec
```

to invoke Aztec C. Refer to your Amiga documentation if you have any problems with the above procedure.

C. Startup Message

When booting from your first floppy disk, you will see the startup message:

```
Welcome to the Wonderful World of Aztec C!!  
Version X.X  
XX/XX/XX
```

The current date and time is requested:

```
Date (MM/DD/YY) ?  
Time (HH:MM:SS) ?
```

Respond to each in the specified format.

You should see a prompt that looks like:

```
1 >
```

This is the standard CLI prompt; it indicates that the CLI is waiting for a command.

LOOKING AROUND

To begin, type the following command to the CLI:

```
info
```

This command displays information about the disks that are currently in the disk drives. It tells you exactly how much space is free on each of your disks.

The Amiga supports what is known as a "hierarchical" file structure that is used to organize files on the disk. An analogy may help to explain this.

Think of a disk as a closet where you want to store different sorts of items. Each disk you have can be thought of as a different closet. If you stuffed all the items you have into this closet, it would become cluttered, making it difficult to find individual items.

So to be more organized, you can let your closet have different sections inside. Now you can have a section labeled "sports" and another labeled "clothes". You can still save other items in this closet as well.

If you go into the "sports" section of your closet, you can have all your sports equipment there, or you can have some more sections *inside* the "sports" section that are labeled "baseball," "football," and "tennis." This could go on for quite a bit, but you should get the idea. With the Amiga, you can create exactly this kind of organization on your disks.

The name you use when talking about a "section" on the disk is a "directory." Type the command:

```
dir
```

This is short for DIRectory and displays the contents of the current directory (or "section") that you are in. You will notice that the first set of files displayed is followed by the word **dir**. This is short for "directory" and indicates that there are some additional directories inside the current directory.

To go into one of these directories, type:

```
cd include
```

where **include** is the name of one of the files in your previous display that was followed by the word **dir**. The **cd** command is short for "Change Directory." Now you are in the include directory and another **dir** command will show you its contents—probably some more directories and some more items.

If you are not sure where you are at any time, type:

```
cd
```

to find your current directory, or location. The first name will be master directory that you are in—usually **df0:** or **df1:**, depending upon which drive you are using. This is followed by the name of each subdirectory that you went through to reach the current directory. Each name is separated from the preceding name by a "/" character.

You can move among directories by using the **cd** command and typing the full name of the directory you want. For example, to go into the **df0:** drive's **c** directory with the first **dir** command, type:

```
cd df0:c
```

If the directory does not exist, an error message is displayed.

Once you are in the **c** directory, try the **dir** command. You will notice that some of the files listed have familiar names like **cd**, **dir**, **format** and **diskcopy**. This is the command directory and is where the CLI gets all the commands that it executes.

To move again, type:

```
cd df0:include/exec
```

This places you into the **exec** directory, which is in the **include** directory, which is on the **df0** disk.

Now, instead of **dir** type:

```
list
```

This command is like **dir**, but it gives more information about each file and directory in the current directory.

A similar command:

```
type errno.h
```

displays the contents of the file **errno.h** on the screen of the Amiga.

The commands **info**, **cd**, **dir**, **list** and **type** are the most often used commands when moving and looking around in the CLI. Try using these commands to look in other directories as well.

WORKING WITH FILES

Move to the top or **ROOT** directory by typing:

```
cd df0:
```

Now, make a directory for your files, to keep them out of the way. Type:

```
makedir test
```

which creates a new directory named **test** in the current directory. Go into **test** by typing:

```
cd test
```

To create a simple program using the **copy** command, type:

```
copy * to hello
```

The `"*"` tells the copy command to get the source from the keyboard and the `to` says to put what it gets into the file `hello`.

The disk whirs and the cursor waits for input without a prompt. Now, type the following short program, exactly as it is written here:

```
main()
{
    printf("hello world!!\n");
}
```

To tell the `copy` command you are done, hold down the `ctrl` key and press the `\` key. You should see the CLI prompt once again.

To make sure the file is correct, use the `type` command again to display the file to the screen. Proof the text that is displayed against the above program; they should be identical.

If you now type `dir`, you will see the file `hello`. Since most C program source files are distinguished by a file name extension of `.c`, use the following command to rename the file:

```
rename hello hello.c
```

This changes the name of the file `hello` to `hello.c`.

A much better way to create and modify files is to use a text editor. The Amiga computer comes with two editors which are described in the *AmigaDOS User's Manual*, `ed` and `edit`. `Ed` is generally considered a much better editor than `edit`, since `ed` is screen edited, while `edit` is line-oriented, which most people find cumbersome. In addition, Aztec C includes a powerful editor named `Z`, which is similar to the UNIX `vi`-editor.

COMPILING, ASSEMBLING, AND LINKING

To make an executable version of your file that you can run, type the following to compile and automatically assemble this program:

```
cc hello.c
```

This invokes the Aztec C compiler to compile the file `hello.c`; it will also automatically assemble the file as well. Do not worry if you receive an

error: illegal character 0x1b (this is caused by the ctrl-\ sequence we used earlier), as it does not affect the code generated.

Do a **dir** command and you will see that a new file, **hello.o**, has been created. This is the object file created by the assembler.

To turn **hello.o** into an executable file, you need to link it with the **c.lib** library. Type:

```
ln hello.o -lc
```

This invokes the linker to link the **hello.o** module with the standard C library and place the output in the file **hello**. Notice how much disk activity takes place during the link phase. The linker is taking all the pieces from the library that it needs and is writing them out to the executable file.

To execute the program, type:

```
hello
```

The words

```
hello world!!!
```

should appear on your screen, followed by a prompt.

To get rid of the **.o** file, type:

```
delete hello.o
```

which deletes the file.

ENVIRONMENT VARIABLES

When the linker links with the libraries, or when the compiler includes header files, you have to tell them where to find them. The easiest way is to use something called an environment variable.

This is a variable with a name whose value is a string and that can be located by any program. Environment variables are created with the **set** command. To see what environment variables currently exist, type:

```
set
```

and the current environment will be displayed.

Notice that two items are defined, CLIB and INCLUDE. These were created by a command in a file whose full name is df0:s/Startup-Sequence. The command in the file looks like:

```
set CLIB=df0:lib/INCLUDE=df0:include
```

The linker uses the CLIB variable to know where to look for the libraries, while the compiler and assembler use the INCLUDE variable to find header files.

THE RAM DISK

When you linked the "hello" program, the linker had to look all over the disk for the parts of the library that it needed and then put them together to make the program. This is why linking took a while. However, you can speed up linking by speeding up the seeking process.

This is accomplished nicely by using a very fast disk or by using a RAM disk. The Amiga computer comes with a built-in RAM disk that lies dormant until you want to use it. Since the RAM disk uses up memory that is also needed for programs to run, you need at least 512K of memory to use it.

To make the link run faster, you first need to copy the library over to the RAM disk. Type:

```
copy df0:lib/c.lib ram:
```

Now, you need to tell the linker where to look, so type:

```
set CLIB=ram:
```

which changes the environment variable used by the linker.

Now if you type:

```
ln hello.o -lc
```

there should be a lot less disk activity, and the linking should be much faster.

Summary

This chapter has described only a few of the many features of the Aztec C development system. From here, you should go on to read the **Compiler**, **Assembler**, and **Linker** chapters of your *Aztec C Reference Manual*.

At some point, you should read the entire *Reference Manual*, as well as the *Aztec C Library Manual*. If you are not already familiar with the AmigaDOS CLI commands and options, it would also be well worth your while to read the *AmigaDOS User's Manual*.

Once you are accustomed to using Aztec C, you can start writing programs that access the special features of the Amiga. These features are discussed briefly in the **Compiler** chapter of your *Aztec C Reference Manual* and in the **Library Functions** chapter of your *Aztec C Library Manual*. We recommend, also, that you purchase the *ROM Kernel* and the *Intuition* manuals, as these books describe all the functions in detail and provide examples of their use.

For more information on these and other Amiga-related books, see the section "Suggestions For Further Reading" that has been included at the front of this manual.

Now that you have completed the **Tutorial**, you should know enough to create some simple programs using Aztec C. As helpful examples, a number of public domain programs have been collected from various sources and included on your Aztec C disks.

Good luck, and enjoy!

Chapter 4 - Technical Support

We have put together a set of guidelines to help you take the most advantage of the technical support service offered by Manx. We ask that you read and follow these guidelines to enable us to continue to give you quality technical support.

Have Everything With You

Try to be organized. When using our phone support, have everything you need with you at the time you call. Our goal is to give you the help you need without keeping you on the phone too long. This can save you a lot of time, and if we can keep the calls as short as possible, we can take more calls each day. This can be to your advantage on days when we are busy and it is hard to get through. Also, have the following information ready when you call technical support. We will ask you for this information first.

- **Your name.** This is necessary in case we need to get back to you with additional information.
- **Phone number.** In case we have additional information we will be able to contact you. This will never be given to anyone, so you need not worry.

- **Name of the product you are using, and its SERIAL NUMBER.**
If you have a cross compiler, please tell us both host and target, even if the problem is with only one side of the system.
- **The revision number of the product you are using.** This should include a letter after the number: i.e., 3.20d or 1.06d. **THIS IS VERY IMPORTANT.** The full version number may be found on your distribution disks or when you run the Compiler.
- **The operating system you are using, and also the version.**
- **The type of machine you are using.**
- **Anything interesting about your machine configuration.** i.e., ram disk, hard disk, disk cache software, etc.

Know What Question You Wish To Ask

If you call with a usage question, please try to have your questions narrowed down as much as possible. It is easier and quicker for all to answer a specific question rather than a general one.

Isolate The Code That Caused The Problem

If you think you have found a bug in our software, try to create a small program that reproduces the problem. If this program is small enough, we will take it over the phone; otherwise we would prefer that you mail it to us, using the supplied problem report, or leave it on one of our bbs systems. Once we receive a "bug report," we will attempt to reproduce the problem and, if successful, we will try to have it fixed in the next release. If we cannot reproduce the problem, we will contact you for more information.

Use Your C Language Book And Technical Manuals First

Manx Technical Support is happy to help you with problems or questions relative to the operation of our products. If you are having difficulty with the C language syntax or a C programming problem in general, please check with a C language programming book, or contact a local university or user group. If you have questions about machine specific code, i.e., interrupts or DOS calls, check with the technical reference manual for that machine and/or its operating system manual.

When To Expect An Answer

A normal turn around time for a question is anywhere from two minutes to 24 hours, depending on the nature of the question. A few questions, like tracing compiler bugs, may take a little longer. If you can call us back the next day, or when the person you speak with in technical support recommends, we will have an in-depth answer for you. But normally we can answer your questions immediately.

Use Our Mail-in Service

It is always easier for us to answer your question if you mail us a letter. (We have included copies of our problem report form for your use.) This is especially true if you have found a bug with our compiler or other software in our package. If you do mail your question in, try to include all of the above information, and/or a disk with the problem. Again, please write small test programs to reproduce possible bugs. See page 4-5 for the Manx mailing address.

Updates, Availability, Prices

If you have questions about updates, availability of software, or prices, please contact the appropriate department. See page 4-5 for a complete listing of the Manx addresses and telephone numbers.

Bulletin Board System

For users of Aztec C, we have a bulletin board system available. See page 4-5 for the different bulletin board numbers.

Follow the questions that will be asked after you are connected. When this is done, you will be on the system with limited access. To gain a higher access level, send mail to SYSOP. Include in this information your serial number and what product you have. Within approximately 24 hours you should have a higher access level, provided the serial number is valid. This will allow you to look at the various information files and upload/download files.

To use the bulletin board best, please do not put large (>8 lines) source files onto the news system, which we use for an open forum question/answer area. Instead, upload the files to the appropriate area, and post a news item explaining the problem you are having. Also, the smaller the test program,

the quicker and easier it is for us to look into the problem, not to mention the savings of phone time. When you do post a news item, please date it and sign it. This will be very helpful in keeping track of questions. Try to do the same with uploaded source files.

Phone Support

And finally, telephone technical support is provided with the purchase of your Aztec C for 90 days from date of purchase. It is available between 10-12 a.m. and 3-5 p.m. eastern standard time. (Times subject to change without notice; for the correct telephone number, see page 4-5.) Phone support is available to registered users of Aztec C.

These guidelines will aid us in helping you quickly through any roadblocks you may find in your development.

Manx Software Systems

Address:

Manx Software Systems
P.O.Box 55
Shrewsbury, NJ 07702

Phones:

Technical Support	(201) 542-1795
Sales (Domestic)	(800) 221-0440
Sales (International)	(201) 542-2121
Updates	(201) 389-0290
FAX:	(201) 542-8386
Bulletin Board: 300/1200 bps (all products)	(201) 542-2793

Note: The above information is subject to change without notice.

MANX PROBLEM REPORT

Date: ____/____/____

Name: _____

Phone #:1-(____)-____-____

Company : _____

Address : _____

Product

C86-PC____ C86-CPM86____

C68k____

C68k-Am____ cII____ c80____

c65-ProDos____ c65-DOS3.3____

cross _____

Version #: _____

Serial #: _____

Op. - sys.: _____

Machine Config.: _____

Description of problem --

Include what has already been attempted to fix the problem and use the reverse side of this sheet if needed.

Index

68020, 1-1

68881, 1-1

A

add-on products, 1-1, 1-3

Amiga 1000, 2-2

Amiga 2000, 2-2

Amiga 2500, 2-2

Amiga 500, 2-2

ANSI standard, 1-6

assembly and C mixed, 2-10

autoboot

Startup-Sequence, 2-21

Aztec C

core systems, 1-1

Developer level, 1-2

Professional level, 1-2

Aztec C add-on products

See add-on products

B

backing up distribution disks, 2-2

bin directory, 2-17

branch shortening, 1-4

bulletin boards, 2-9

C

C and assembly mixed, 2-10

C directory, 2-17

C libraries, 2-17

c.lib, 2-17, 3-7

cc command, 3-6

cd command, 3-4

CLI

default environments, 3-1

loading into RAM, 2-16

- makedir, 2-19
- Preferences program, 2-20
- prompt, 3-3
- run command, 2-17
- using CLI, 3-2
- CLI commands
 - cd, 3-4
 - copy, 3-5 - 3-6
 - delete, 3-7
 - dir, 3-4
 - info, 3-3
 - list, 3-5
 - rename, 3-6
 - type, 3-5
- CLI icon, 3-2
- CLIB environment variable, 3-8
- compile, 3-6
- components, Aztec C
 - debuggers, 2-9
 - files, 2-4 - 2-5
 - libraries, 2-5 - 2-6
 - pragmas, 2-8
 - pre-compiled header files, 2-8
 - resident libraries, 2-7
- copy command, 3-5 - 3-6
 - example, 3-8
- core systems, 1-1 - 1-2
- cross development, 1-3

D

- data definitions, 2-5 - 2-6
- debuggers, 2-9
- delete command, 3-7
- Developer System, 1-2
- development environments
 - choice of, 3-1
- device drivers, 1-1
- devs directory, 2-17
- diff, 1-2
- dir command, 3-4

- directory
 - command, 3-4
 - definition, 3-4
 - display current, 3-4 - 3-5
- Diskcopy command, 2-16
- documentation, 1-1, 1-3
 - reading order, 1-4 - 1-5
- draft-proposed ANSI, 1-6

E

- ed, 3-6
- edit, 3-6
- editor, 3-6
- environment variables, 2-7, 3-7
 - CCEDIT, 2-11 - 2-13
 - CCOPTS, 2-11, 2-13
 - CLIB, 3-8
 - INCLUDE, 3-8
- executable file, 3-7

F

- file structure, 3-3
- file types, 2-4 - 2-5
- floating point libraries, 2-14, 2-17 - 2-18
 - 16 bit int library, 2-17
 - 32 bit int library, 2-18
 - 64 bit int library, 2-18
 - 68881 coprocessor, 2-18
 - Amiga IEEE, 2-18
 - large code/large data , 2-17
 - Manx IEEE, 2-18
 - Motorola Fast Floating Point, 2-18
 - small code/small data , 2-17
- Format command, 2-16
- freeware, 2-9
- functions.h, 2-5 - 2-8, 2-13
 - hi compiler option, 2-8

G

grep, 1-2

H

hard disk, using Aztec C with, 2-19

I

include directory, 2-20

INCLUDE environment variable, 3-8

info command, 3-3, 3-5

installation procedure, 1-4, 2-3, 2-14

 hard disk, 2-19 - 2-21

 one floppy drive, 2-18

 two drives + 1 MB memory, 2-15 - 2-16

 two drives + less than 1 MB memory, 2-16 - 2-17

 two drives, quick installation, 2-19

Intuition, 1-1

L

large code/large data, 2-17

lib directory, 2-17

library source, 1-3

libs directory, 2-17

link, 3-7

linking, 3-7 - 3-8

lint directory, 2-19

list command, 3-5

M

m.lib, 2-17

make, 1-2

makedir command, 2-19, 2-21

math libraries, 2-17

 See also floating point libraries

multiple segment support, 2-10

N

notation conventions, 1-5

O

- object file, 3-7
- overlay support, 2-10

P

- pragmas, 2-8
- pre-compiled header files
 - hi compiler option, 2-8
- Preferences program, 2-20, 3-2
- Professional System, 1-2

Q

- quick installation, 2-19
- QuikFix, 2-10
 - qf compiler option, 2-11
 - wq compiler option, 2-12
 - AREXX, 2-11
 - batch processing, 2-12
 - further information, 2-13
 - make utility, 2-13
 - Z editor commands, 2-11 - 2-12
 - Z editor options, 2-11

R

- RAM disk, 2-15, 3-8
 - system requirements, 3-8
- RAM loading, 2-16
- reading order, 1-4
- README file, 1-4, 2-2 - 2-3
- resident libraries
 - Amiga Workbench, 2-7
 - user created, 2-8
- root directory
 - defined, 3-5
- Run command, 2-17, 2-19

S

- sdb, 1-3, 2-2
- set command, Amiga CLI, 2-7
- set command, Aztec, 2-7, 2-20, 3-7
- shareware, 2-9
- small code/small data, 2-17
- source level debugger, 1-3, 2-2
 - See also sdb
- starting Aztec C
 - with floppy disks, 3-2
 - with hard disk, 3-2
- startup message, 3-3
- Startup-Sequence, 2-16
- system overview, 2-3
- system requirements, 2-1 - 2-2, 2-15 - 2-16

T

- technical support, 4-1
 - bulletin board system, 4-3
 - mail-in service, 4-3
 - phone hours, 4-4
 - phone number, 4-4
 - phone support, 4-4
 - procedures, 4-2 - 4-3
 - required info, 4-1 - 4-2
 - updates, 4-3
- text editor, 3-6
- third party software, 2-9
- transcendental functions, 2-17

U

- UniTools, 1-2, 1-5
- UNIX, 1-1 - 1-2, 3-6

W

- write protect tabs, 2-15

Z

- z editor, 1-2, 3-6